A METHOD FOR TRAINING FEED FORWARD NEURAL NETWORK TO BE FAULT TOLERANT

H. Elsimary*, S. Mashali*, S. Shaheen**
* Electroninces Research Inst., Cairo, Egypt
** Faculty of Engineering,Cairo University, Cairo, Egypt
Email : Hamed@EGFRCUVX.Bitnet

Abstract

This paper describes a method for training feed forward neural network to be fault tolerant against the weight perturbation. Our measure for fault tolerance is the deviation of the network's output after training, when each interconnection weight is perturbed, from that output without perturbation. In our method we try to maintain that deviation as minimum as possible. We have used this measure because it can represent that kind of error arises when neural networks are implemented in hardware.

Introduction

As Artificial Neural Networks (ANNs) has shown great performance in many fields, and has shown its abilities in solving problems difficult to be solved by conventional methods of computation, as it is continually progressing and its application is wide spreading, it is important to study and insure the reliability of its function under different working conditions.

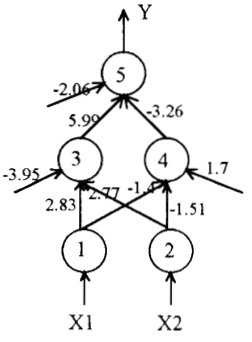
One of the advantages of ANNs is that the processing is distributed among many nodes, this provides a good degree of fault tolerance and graceful degradation to the system even if some of its nodes or interconnection fails to function properly. But this features needs to be emphasized, improved, and maximized through different techniques. Many researches has been done to study, analyze, and improve the fault tolerance of ANNs in the functional level, but little however have not been adreesed that fault related to the hardware implementation [2]. There are many reliability modeling and improvement techniques exist for conventional computers, these models are developed to predict the reliability and fault tolerance characteristics of each functional block of the system such as redundancy and self checking. But the issue is different in ANNs than that of conventional computer architecture, therefore a different approach is required. In this work we will introduce a method for Improving the fault tolerance of ANNs, in the form of a feed forward model, since the feed forward model with the very famous backpropagation algorithm has become very popular and has been used in different engineering applications [3]. We have used a suitable definition and criteria for fault tolerance measurement that can deal with faults that arise when the ANNs implemented in hardware [4,5], and by using this measure an algorithm for training the ANN subject maximizes the fault tolerance has been developed.

Modeling faults in ANNs

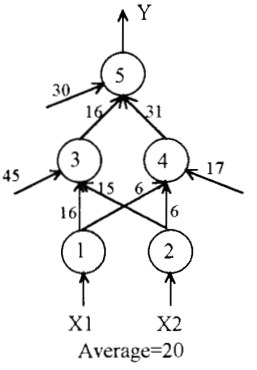
Taking into considerations the hardware implementation of ANNs, whether this implementation is analog or digital [4,5] with limited numerical precision [6,7], we should have a model that can deal with faults that comes from each way of implementation.

0-7803-1363-1/93 $3.00 © 1993 IEEE

specific number of iterations, and in fig (3b) represents the percentage deviation of the network's output when the particular weight or threshold is perturbed by a value of +/- ΔW (in this case ΔW=10%). for example the number in the connection from the first unit 3 in the hidden layer to the unit 5 in the output layer, means that the output will be changed by a factor of 47% if this interconnection weight is perturbed by ΔW and averaged all over the training exemplars. The average network sensitivity to the weight perturbation is 35 in this case.



Fig(4a)



Fig(4b)

On the other hand the network of fig(4) has been trained with initial weight set W2 (not shown also) which leads to a final state shown in fig(4a), the percentage output change when each individual weight or threshold is perturbed by Δw is shown in fig(4b) the average fault tolerance figure has been improved but we don't know yet whether this is the optimum or not, so the main concern right now is to find the optimum weight set that maximizes the fault tolerance. the following example will explain in more details the idea of the fault tolerance improvement.

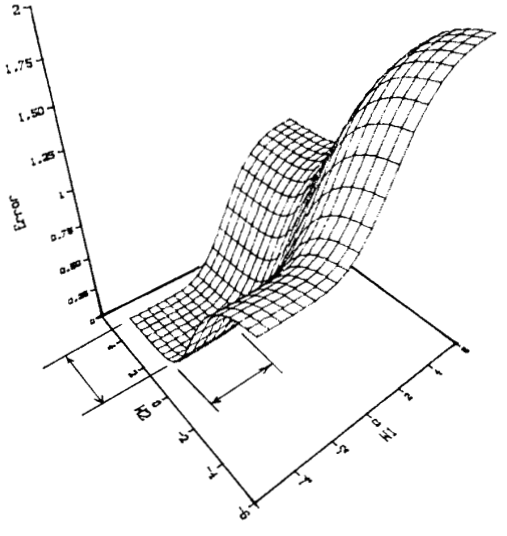
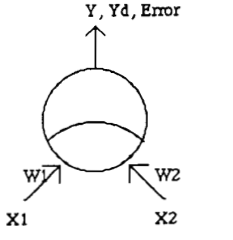


Fig (5a)



X1	X2	Yd
0	1	0
1	0	1

Training Pattern

Fig (5b)

A single neuron is being trained using them training pattern shown in fig(5b), the output of the neuron is determined by the equation:

$$Y = \frac{1}{1+e^{-Net}} \quad ; \text{Where } Net = \sum w_i x_i$$

And resulting Error over all the training patterns is determined by:

$$E = 1/T \sum (Y - Y_d)^2 \quad ; \text{Where T is the number of training patterns}$$

This example has been considered for the sake of the ability to imagine and draw the error as a function of weight vector. The error surface in our example as a function of w1, and w2 is shown in fig (5a). The objective of any training algorithm is to find the point that minimizes the error function, in this specific case there are many points that can achieve minimum error as shown in the marked region in fig (5b). Selecting one of these points is the role of our algorithm, among these points there is one that can improve the fault tolerance characteristics of the neuron, i.e., decreases its sensitivity to the weight perturbation.

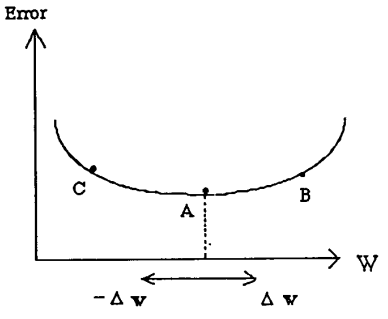


Fig (6a)

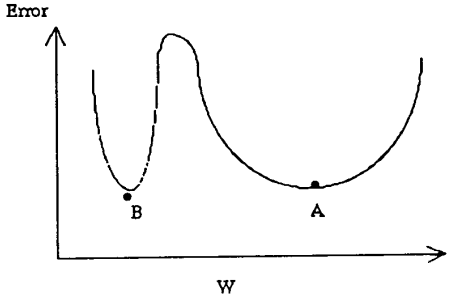
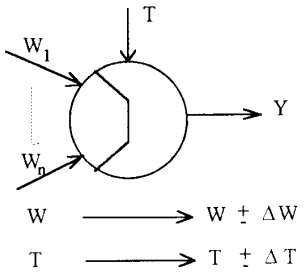


Fig (6b)

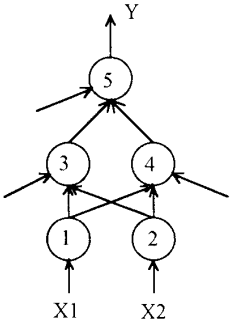
As shown in fig (6a), point A is considered better than point B or C in the sense of fault tolerance, as the resulting error will not be much affected if the weight is perturbed by ΔW. Also there may be, in other cases, many minima but one of them may be better than the other in the sense of fault tolerance, as shown if fig (6b) point A is considered better than point B for the same reason. In multidimensional surfaces where the number of variables is more than two, it will be difficult to imagine or draw the error surface, but there may be many minima, where one of them could be better than the other in the sense of our measure of the fault tolerance.

From previous discussion we have shown that the same network can have different fault tolerance characteristics if with different initial conditions, this will lead us to develop an algorithm for training the ANN not only to minimize the output error but also to maximize the fault tolerance capability. We did not discuss in the previous discussion whether the output change will affect the desired output values or not, so we will come to a formal definition of the problem for fault tolerance maximization that will be discussed in the following section.



Fig(1)

A possible method to model the faults for ANNs could be the deviation from the original if the interconnection weights and threshold values have been perturbed after the network has been trained. Roughly speaking the ANN should still give correct or at least near correct results even with the existence of fault, the interconnection weight is being perturbed by an amount +/- ΔW as shown in figure (1), this will be illustrated by an example.

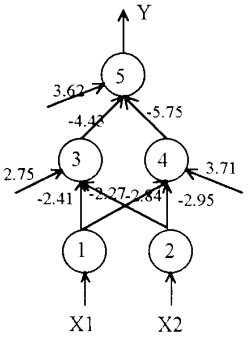


X1	X2	Yd
0	0	0
0	1	0
1	0	0
1	1	1

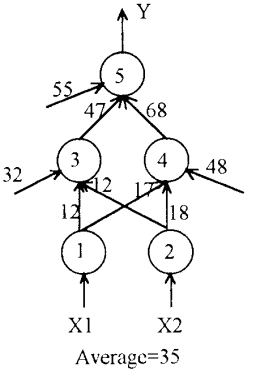
Training Pattern

Fig (2)

Consider the ANN of fig (2) trained to simulate the logic AND function. It has been noticed that when the network is trained with the same algorithm (Backward Error Propagation), but starting from different initial weight set and allowed to train until certain error limit, it will settle down at different final states that leads to different fault tolerance degrees.



Fig(3a)



Fig(3b)

For example the final state of a network trained at initial weight set W1(only the final state is shown) shown in fig (3a, 3b), each number in fig(3a) represents the weight values after a

Problem Definition and Solution Approach

The problem we are addressing is to find the weight matrix for a fully connected network, each neuron has a sigmoid transfer characteristics that will minimize the total error and lead to a better fault tolerance characteristics. In our definition to the fault tolerance, the network should still give correct or near correct result even with the interconnection weights are perturbed with $+/-\Delta W$.

ANN is fault tolerant if the computations performed by N', which is a network derived from N by perturbing the interconnection weights by $+/-\Delta W$ is close to the computations performed by N.

E(W) ; Error function, deviation of the actual output of network N from the desired

E'(W') ; Error function of network N' with weight matrix W' Error is being computed for single weight perturbation and averaged all over the interconnection weight

The problem will be formulated as :

$$\min_{w} E(W)$$

subject to : $E'(W')-E(W) < \varepsilon$ where ε is a relatively small number

This a nonlinear optimization problem with multivariants and nonlinear constraints. The problem has been solved using a standard subroutine from the math library which uses successive quadratic programming and finite difference method for solving such optimization problems [8]. Simulation result for training a network to map the logical 'AND' function will be shown next. The training program has been allowed to search for the optimum weight matrix that minimizes the total error subject to be fault tolerant against perturbation in the interconnection weights of +/- 10%, and allowed to search for the weight values in the range -10 to 10, and allowed for a maximum error deviation of 0.000005, the results are summarized as follows:

ε : 0.00005

Perturbation : 10%

Weight matrix:

T3	W31	w32	T4	w41	w41	T5	w53	w54
-0.43	-9.58	3.25	8.92	-4.34	-9.63	4.90	-9.63	-9.96

This means that this network will withstand to a weight change up to 10% with maximum deviation from the actual output of 0.00005.

Conclusion

It has been shown that the same network has different fault tolerance characteristics depends on the way the network is trained. We has used a suitable fault model that can deal with faults that arises with hardware implementation. We have used this fault model to develop a training method for the ANNs to improve its fault tolerance capabilities and demonstrated by examples.

References:

[1] Eva Koscienly-Bunde, "Effect of Damage in Neural Networks", Journal of statistical Phys., Vol. 58, Mar. 1990.

[2] R. K. Chun, L. P. McNamee, "Immunization of Neural Networks Against Hardware Faults", IEEE Int International Symposium on Circuits and Systems, vol 1 pp. 714-718, 1990.

[3] James L. McClelland, and David E. Rumelhart, "Exploration in Parallel Distributed Processing", MIT Press 1988.

[4] Carver Mead, "Analog VLSI and Neural Systems", Addison Wesley, 1989.

[5] Ulrich Ramucher, Ulrich Ruckert, "VLSI Design of Neural Networks", Kluwer Academic Publisher, 1991.

[6] Markus Hoefeld and Scott E. Fahlman, "Learning with limited Numerical Precision Using the ascade-Correlation Algorithm", IEEE Trans. on Neural Networks, Vol. 3, No. 4, pp. 602-611, July 1992.

[7] P.W. Hollis, J.S. Harper, and J.J. Paulos, "The effect of Precision Constraints in Backpropagation learning network", Neural Computation, vol. 2, pp. 363-373, 1990.

[8] Schittkowski, K. , "Nonlinear Programming Codes, Lecture Notes in Economics and mathematical Systems", Springer-Verlag, Berlin, Germany, 1980.